*Process Manager Documentation*

A process is the unit of work in a modern time-sharing system.

A system therefore consists of a collection of processes: operatingsystem processes executing system code and user processes executing user code.

A batch system executes jobs, whereas a time-shared system has user programs, or tasks.

The terms job and process are used almost interchangeably in this text.

Informally, as mentioned earlier, a process is a program in execution. A process is more than the program code, which is sometimes known as the text section. It also includes the current activity, as represented by the value of the program counter and the contents of the processor’s registers. A process generally also includes the process stack, which contains temporary data (such as function parameters, return addresses, and local variables), and a data section, which contains global variables. A process may also include a heap, which is memory that is dynamically allocated during process run time. The structure of a process in memory is shown in Figure 3.1.

We emphasize that a program by itself is not a process. A program is a passive entity, such as a file containing a list of instructions stored on disk (often called an executable file). In contrast, a process is an active entity, with a program counter specifying the next instruction to execute and a set of associated resources.

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. A process may be in one of the following states: • New. The process is being created. • Running. Instructions are being executed. • Waiting. The process is waiting for some event to occur (such as an I/O completion or reception of a signal). • Ready. The process is waiting to be assigned to a processor. • Terminated. The process has finished execution

• Process state. The state may be new, ready, running, waiting, halted, and so on. • Program counter. The counter indicates the address of the next instruction to be executed for this process. • CPU registers. The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward (Figure 3.4). • CPU-scheduling information. This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters. (Chapter 6 describes process scheduling.) • Memory-management information. This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system (Chapter 8).

• Accounting information. This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on. • I/O status information. This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

The process model discussed so far has implied that a process is a program that performs a single thread of execution. For example, when a process is running a word-processor program, a single thread of instructions is being executed. This single thread of control allows the process to perform only one task at a time.

**System.Diagnostics**

Performance counter

The counter is the mechanism by which performance data is collected. The registry stores the names of all the counters, each of which is related to a specific area of system functionality. Examples include a processor's busy time, memory usage, or the number of bytes received over a network connection.

Each counter is uniquely identified through its name and its location. In the same way that a file path includes a drive, a directory, one or more subdirectories, and a file name, counter information consists of four elements: the computer, the category, the category instance, and the counter name.

# Microsoft.VisualBasic.Devices